REMARKS:

Claims 1-55 were presented for examination and were pending in this application. In an Official Action dated April 13th, 2006, claims 1-55 were rejected. Applicants thank Examiner for examination of the claims pending in this application and address Examiner's comments below.

Applicants herein amend claims 17 and 19. These changes are believed not to introduce new matter, and their entry is respectfully requested. The claims have been amended to expedite the prosecution of the application in a manner consistent with the Patent Office Business Goals, 65 Fed. Reg. 54603 (Sept. 8, 2000). In making these amendments, Applicants have not and do not narrow the scope of the protection to which Applicants consider the claimed invention to be entitled and do not concede that the subject matter of such claims was in fact disclosed or taught by the cited prior art. Rather, Applicants reserve the right to pursue such protection at a later point in time and merely seek to pursue protection for the subject matter presented in this submission.

Based on the above Amendment and the following Remarks, Applicants respectfully request that Examiner reconsider all outstanding rejections, and withdraw them.

Response to Rejection Under 35 USC 103

In the 4th paragraph of the Office Action, claims 1, 29-32, and 42-46 were rejected under 35 USC § 103 as allegedly being obvious over U.S. Patent No. 6,542,991 to Joy el at ("Joy") in view of U.S. Patent No. 6,317,774 to Jones et al ("Jones"). This rejection is respectfully traversed.

Independent claim 1 recites:

> a hardware thread scheduler for identifying which of said program threads said processor executes and configurable to allocate available processing time of the processor among at least the first and second program threads by causing thread-switching at a fixed time according to a predetermined fixed schedule, said schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles.

Similarly, independent claim 46 recites:

> "A computer based method for switching between program contexts in a multithreading pipelined processor having a hardware thread selector and an execution pipeline, the method comprising:...
> switching the processor from the first thread state to the second thread state by coupling the execution pipeline from the first set of data storage devices to the second set of storage devices via the hardware thread selector at a fixed time according to a predetermined fixed execution schedule, said execution schedule specifying that the processor should switch to the first thread state every first number of cycles and that the processor should switch to the second thread state every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles."

The hardware thread scheduler is configurable to cause the processor to switch from one thread to another thread at a fixed time according to a predetermined fixed schedule. This is greatly beneficial because it provides a predictable execution time for threads.

Joy discloses "oblivious thread switching", in which the thread executed by the processor is switched every N cycles. (Col. 17, ln. 1-4) In the oblivious thread switching disclosed by Joy, a first thread would be executed every K cycles (where K is N times the number of threads in the oblivious switching rotation), a second thread would be executed

every K cycles, and so on. As correctly noted by the Examiner, Joy does not teach thread-switching at a fixed time according to a predetermined fixed schedule.

The deficiencies of Joy are not rectified by the disclosure of Jones. Jones merely describes a multithreading operating system scheduling feature using a directed acyclic graph of nodes (col. 2, lines 51-54). Jones discloses a method for determining at a software level which application to execute when the processor becomes available (see Abstract). Applications can request reservations, which specify a number of time units for which the application will be executed. Applications can also specify reservation windows, which specify that the application will execute at least every number of time units (see col. 5, lines 8-17).

The combination of Joy and Jones, as suggested by the Examiner, is deficient at least for failing to disclose the hardware thread scheduler of the claim. Jones and Joy, both alone and in combination, fail to disclose a hardware thread scheduler configurable to allocate available processing time among a first and a second program thread according to a predetermined fixed schedule, said schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles. While Joy may disclose a hardware thread scheduler, the hardware thread scheduler of Joy, as admitted by the Examiner, is explicitly not configurable to allocate available processing time in the manner described in the claim. Jones, describing a software-based activity scheduler, does not disclose any hardware thread scheduler whatsoever, much less a hardware thread scheduler configurable as in the claim.

Jones fails to disclose a schedule specifying that a first thread should be allocated processing time every first number of cycles and that a second thread should be allocated processing time every second number of cycles. The reservations of Jones are given in "arbitrary time units" (see col. 6, line 67), which roughly specify the percentage of total processor time and the general time frame during which an application will run (see col. 5, lines 35-37). Jones makes no suggestion that a time unit could correspond to a fixed number of cycles. In fact, a time unit in Jones *cannot* correspond to a fixed number of cycles, because, as a software program executing on a conventional CPU, the software scheduling unit of Jones cannot predict and does not control the number of cycles that will be lost in determining and performing a switch from one application to the next. As software, the application scheduler of Jones must itself be executed on the processor to determine activity switches. Jones teaches that the application scheduler executes for a "bounded amount of time" (see Col. 2, lines 30-34) during which the application scheduler consumes cycles. The imprecise resolution of software-based scheduling and the need to execute the software application scheduler itself render Jones incompatible with cycle-specific scheduling. A reservation indicating that an application should be executed every 100 time units, for example, could not indicate a specific number of cycles, because any varying number of cycles could be lost in the execution of the application scheduler and in performing the switch to once a determination has been made. The software-based application scheduler of Jones assigns processing time and executes at too high a level to allocate specific numbers of instruction cycles.

To establish a prima facie case of obviousness, each and every element of the claimed invention must be taught or suggested by the references. However, the combination of Joy

and Jones has been shown to be deficient for failing to disclose either "a predetermined fixed schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles" or "a hardware thread scheduler...configurable to allocate available processing time of the processor among at least the first and second program threads by causing thread-switching at a fixed time" according to such a schedule.

Furthermore, the proposed modification of a reference cannot change the principle of operation of a reference (see MPEP 2143.01). The only *scheduled* thread switching taught by Joy is the oblivious thread switching, which switches threads every N cycles. In the oblivious thread switching of Joy, "individual flip-flops locally determine a thread switch without notification of stalling" (see col. 3, lines 33-35) and is the switching is "typically implemented using a simple counter for counting cycles between switches" (see col. 7, lines 4-5). Joy teaches oblivious thread switching determined by low-level hardware circuits. The flip-flop-implemented thread switch of Joy is principally incompatible with Jones, which requires execution of a software scheduler to determine application switches. The Examiner has cited no art explaining how the individual flip-flops of Joy could be modified, for example, to construct a scheduling graph, to evaluate and in some cases refuse reservation requests, to traverse nodes, to add nodes to the scheduling graph, or to perform the various other tasks required to implement the activity scheduling of Jones. To implement a hardware thread scheduler capable of executing the software methodology of Jones would require a substantial reconstruction and redesign of the hardware shown in Joy, as well as changes in the basic principle under which Joy was designed to operate.

For at least these reasons, Applicants submit that the rejection of claims 1 and 46 under 35 USC 103 is improper and should be withdrawn.

As claims 29-32 and 42-45 are dependent from claim 1, all arguments presented with regard to claim 1 are hereby incorporated so as to apply to claims 29-32 and 42-45. For at least these reasons, Applicants submit that the rejection of claims 29-32 and 42-45 should be withdrawn.

In the 15th paragraph of the Office Action, claims 2-3, 13-17, 19, and 21-24 were rejected under 35 USC § 103 as allegedly being obvious over Joy in view of U.S. Patent No. 6,493,741 to Emer et al ("Emer"). This rejection is respectfully traversed in view of the amended claims.

Independent claim 17, as amended, recites:

> "A computer based system for switching between program contexts comprising:
>
> a pipelined processor capable of having a first program thread and a second program thread in an execution pipeline having a thread selection hardware configured to switch program threads within an instruction cycle;...
>
> a hardware thread scheduler for identifying which of said program threads said processor executes and configurable to allocate available processing time of the pipelined processor among at least the first and second program threads according to an execution schedule;
>
> wherein said thread selection hardware in the pipelined processor switches from said first thread state to said second thread state between consecutive instruction cycles in response to the hardware thread scheduler identifying which of said program threads said processor executes."

Similarly, independent claim 19, as amended, recites:

> switching the processor from executing the first program thread to executing the second program thread ~~between the end of~~ within an execution

cycle and before the beginning of a next consecutive execution cycle by coupling the execution pipeline from the first set of data storage devices to the second set of storage devices via the hardware thread selector.

Switching program threads within an instruction cycle and switching thread states between instruction cycles beneficially allows switching between one program context and another without incurring any time penalty. Switching from one thread to another within an execution cycle and before the beginning of a next consecutive execution cycle beneficially allows switching to occur without the loss of any execution cycles.

It is important to note that switching threads "between consecutive instruction cycles" does not specify how frequently thread switches occur, but rather how quickly the thread selection hardware executes those switches in response to the hardware thread scheduler. Thus, while the claim certainly includes the case of in which switching occurs every cycle, it should not be misunderstood as being limited to that particular case. Fundamentally, the question of *how long* a thread switch takes in response to a hardware thread scheduler should not be confused with the question *how often* a thread switch occurs. A disclosure of switching threads at every instruction cycle would not be sufficient to anticipate switching threads *between* consecutive instruction cycles.

Joy discloses "oblivious thread switching", in which the thread executed by the processor is switched every N cycles. (Col. 17, ln. 1-4) But this disclosure is an answer to the question of how often a thread switch occurs, and is not relevant to the claim element in question. The fact that a processor switches threads every N cycles has no bearing on the cost associated with those switches.

Joy teaches that "the thread switch logic supports a fast thread switch with a very small delay, for example three cycles or less." (Col. 16, ln. 61-62) Joy goes on to disclose an even more difficult constraint, "The thread switch logic generates the TID signal with a thread switch delay or overhead of one processor cycle." Thus, Joy discloses a very small delay in switching, but nonetheless Joy discloses a delay. Even the most ambitious portions of Joy teach the need of an overhead of at least one processor cycle. Joy is fundamentally incompatible with switching without a time penalty, as Joy explicitly discloses that switching *between consecutive instruction cycles* in response to the hardware thread scheduler is not possible.

The deficiencies of Joy are not rectified by Emer. Emer discloses a multithreaded architecture in which thread switching occurs every cycle. As discussed above, the frequency of thread switching should not be confused with the overhead of thread switching. While Emer may disclose switching every cycle, Emer does not disclose "switching between consecutive instruction cycles" *and* "switching within an instruction cycle".

Emer acknowledges that switching overhead has not been completely eliminated in the disclosed multithreaded architecture: "Multithreaded processors better tolerate long-latency operations, *effectively* eliminating vertical waste." (col. 1, ln. 58-60, emphasis added) As vertical waste has been only effectively eliminated, Emer suggests that in fact some number of cycles goes completely unused in the process of switching from one thread to another. (See Emer's definition of vertical waste, col. 1, ln. 41-42.) Switching between consecutive cycles requires that no cycles go unused in the process of switching from one thread to another.

Admitting the presence of vertical waste in Emer, the Examiner provides an alternate explanation, asserting that "the only way vertical waste can exist... is if there [are] less total threads than cycles in a latency event". It is not clear where Emer states that this is the only way vertical waste can exist. In fact, this particular example is neither illustrated in the figures nor discussed by Emer. While it is possible that this scenario could be one example of vertical waste, it is reading beyond the disclosure of Emer to assume that it is the *only* exception, and that otherwise vertical waste would be completely eliminated. It is known in the art that switching threads typically results in some number of cycles going unused, and Emer admits that vertical waste has been only effectively eliminated. Therefore, absent a disclosure explicitly asserting that no cycles go unused in the switching process, one of skill in the art would understand that in the system disclosed by Emer, at least one cycle goes unused during the switching process.

As for the definition of "processor cycle", if the "processor cycle" disclosed by Emer is understood to mean a regularly occurring period of time, then Emer does not disclose switching "between consecutive instruction [execution] cycles", as vertical waste has only been "effectively" eliminated and at least some number of processor cycles goes unused between switches. If the "processor cycle" disclosed by Emer is understood to mean the period of time that each thread actually executes, as suggested by the Examiner, then Emer does not disclose switching "within an instruction [execution] cycle", because such a processor cycle would not include any amount of time for switching. Because Emer has not taught zero-time context switching, switches take some real amount of time, and therefore it

would be impossible for switching to occur both between consecutive cycles and within a cycle.

To establish a prima facie case of obviousness, each and every element of the claimed invention must be taught or suggested by the references. However, the combination of Joy and Emer has been shown to be deficient for failing to disclose a "pipelined processor configured to switch program threads within an instruction cycle...and...thread selection hardware in the pipelined processor switches from said first thread state to said second thread state between consecutive instruction cycles" and for failing to disclose "switching the processor from executing the first program thread to executing the second program thread within an execution cycle and before the beginning of a next consecutive execution cycle". For at least these reasons, Applicants submit that the rejection of claims 17 and 19 under 35 USC 103 is overcome and should be withdrawn.

As claims 2-3 and 13-16 are dependent from claim 17, and claims 21-24 are dependent from claim 19, all arguments presented with regard to claims 17 and 19 are hereby incorporated so as to apply to claims 2-3, 13-16 and 21-24. For at least these reasons, Applicants submit that the rejection of claims 2-3, 21-24, and 13-16 should be withdrawn.

In the 28th, 30th, and 37th paragraphs of the Office Action, Examiner rejects claims 4-12, 18, 20, 25-28, under 35 USC § 103(a) as allegedly being unpatentable in view of Joy, Emer, and various combinations of U.S. Patent No. 6,567,839 to Borkenhagan et al. ("Borkenhagan"), U.S. Patent No. 6,085,215 to Ramakrishnan et al. ("Ramakrishnan"), U.S. Patent No. 6,026,503 to Gutgold et al. ("Gutgold").

Similarly, in the 41st, 43rd, and 54th paragraphs of the Office Action, Examiner rejects claims 33-41, and 47-55 under 35 USC § 103(a) as allegedly being unpatentable in view of Joy, Jones, and various combinations of Borkenhagen, Ramakrishnan, and Gutgold. These rejections are respectfully traversed.

As claims 4-12 and 18 are dependent on claim 17, all arguments advanced above with respect to claim 17 are hereby incorporated so as to apply to claims 4-12 and 18. As claims 20, and 25-28 are dependent from claim 19, all arguments advanced above with respect to claim 19 are hereby incorporated so as to apply to claims 20, and 25-28.

Claim 17 and its dependent claims have been shown to be patentable over the combination of Joy and Emer, at least because the combination fails to disclose "pipelined processor configured to switch program threads within an instruction cycle...and...thread selection hardware in the pipelined processor switches from said first thread state to said second thread state between consecutive instruction cycles." Similarly, Claim 19 and its dependent claims have been shown to be patentable over the combination of Joy and Emer, at least because the combination fails to disclose "switching the processor from executing the first program thread to executing the second program thread within an execution cycle and before the beginning of a next consecutive execution cycle." These deficiencies are not remedied by the various disclosures, either alone or in combination, of Borkenhagen, Ramkrishnan, and Gutgold.

As claims 33-41 are dependent on claim 1, all arguments advanced above with respect to claim 1 are hereby incorporated so as to apply to claims 33-41. As claims 47-55

are dependent from claim 46, all arguments advanced above with respect to claim 46 are hereby incorporated so as to apply to claims 47-55.

Claim 1 and its dependent claims have been shown to be patentable over Joy and Jones, at least because the combination fails to disclose "a hardware thread scheduler for identifying which of said program threads said processor executes and configurable to allocate available processing time of the processor among at least the first and second program threads by causing thread-switching at a fixed time according to a predetermined fixed schedule, said schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles." Similarly, Claim 46 and its dependent claims have been shown to be patentable over Joy and Jones, at least because the combination fails to disclose "switching the processor from the first thread state to the second thread state by coupling the execution pipeline from the first set of data storage devices to the second set of storage devices via the hardware thread selector at a fixed time according to a predetermined fixed execution schedule, said execution schedule specifying that the processor should switch to the first thread state every first number of cycles and that the processor should switch to the second thread state every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles." These deficiencies are not remedied by the various disclosures, either alone or in combination, of Borkenhagen, Ramkrishnan, and Gutgold.

Examiner has cited Borkenhagen as allegedly teaching a processor switching between states by changing a state selection register. Borkenhagen discloses switching threads in

response to a cache miss or external interrupt signal (Col. 6, ln. 22-42), said switches incurring the conventional "latency and performance penalties associated with switching threads." (Borkenhagen, col. 15, ln . 37-48). Borkenhagan does not disclose the deficiencies of Joy or the deficiencies of Joy in combination with Emer cited above.

Likewise, Ramakrishnan is cited to make up for Joy's lack of "thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread" limitation. Ramakrishnan simply describes a software scheduler that uses a round robin approach to thread scheduling using conventional context switching. See Ramakrishnan, col. 9, ln. 9-10. The switching in Ramakrishnan, like in Borkenhagen, is conventional context switching that involves "an associated overhead in invoking the new thread." (Ramakrishnan, col. 12, lines 61-62, see also, col. 13, lines 6-7 "avoid time consuming context switching"). Ramakrishnan does not disclose the deficiencies of Joy or the deficiencies of Joy in combination with Emer cited above.

Finally, Gutgold is relied upon to provide the different access speed memory devices recited in claims 10-12 and 39-41 that are not explicitly described in Joy, Ramakrishnan, or Emer. However, the combined reference still fails to teach or suggest the fixed time or consecutive-cycle context switching recited in the claims. Gutgold does not describe any context switching. Aside from the fact that Gutgold describes a microprocessor controlled system and associated microprocessor system components, Applicants see no other relation to Applicants' invention.

Accordingly, for at least the reasons set forth above, Applicants submit that claims 4-12, 18, 20, 25-28, 33-41, and 47-55 are patentable over the cited combined references and request that these rejections be withdrawn.

Conclusion

In sum, Applicants respectfully submit that claims 1 through 55, as presented herein, are patentably distinguishable over the cited references. Therefore, Applicants request reconsideration of the basis for the rejections to these claims and request allowance of them.

In addition, Applicants respectfully invite Examiner to contact Applicants' representative at the number provided below if Examiner believes it will help expedite furtherance of this application.

Respectfully Submitted,
NICHOLAS J. KELSEY, ET AL.

Date: 6/13/06 By: _____

Hector J. Ribera, Attorney of Record
Registration No. 54,397
FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
Phone: (650) 335-7192
Fax: (650) 938-5200
E-Mail: hribera@fenwick.com